Exhibit B

Amendments to Claims

Please amend the claims as indicated in the listing below, which listing supercedes and replaces all prior listings of claims:

1. (Currently Amended) A method of operation of a control system, comprising

A. executing a first sequence of instructions in any of a first computer process and first computer thread, ~~(collectively, "first process")~~ collectively, "first process,

B. executing a second sequence of instructions in any of a second computer process and second computer thread, ~~(collectively, "second process")~~ collectively, "second process", the second process being loosely coupled with respect to the first process,

C. comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence,

D. responding to failure of the first and second processes to achieve comparable states by rolling back each of the first and second processes to prior states in which a favorable comparison was achieved, and

~~E.~~D. wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, a process control enterprise server, an industrial control device, an industrial control system, an environmental control device, an environmental control system, other control device, and other control system.

2. (Original) A method according to claim 1, comprising the step of executing step (C) one or more times over a time interval in order to determine whether the first and second processes achieve comparable states following completion of execution of the first instruction sequence by the first process.

3. (Original) A method according to claim 2, comprising the step responding to a favorable comparison in step (C) by repeating steps (A) – (C) with a third instruction sequence in place of

the first instruction sequence, and with a fourth instruction sequence in place of the second instruction sequence.

4. (Original) A method according to claim 3, comprising the steps of

selecting the third instruction sequence as a function of a state of the first process following execution of the first instruction sequence, and

selecting the fourth instruction sequence as a function of a state of the second process following execution of the second instruction sequence.

5. (Original) A method according to claim 4, comprising comparing a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

6. (Currently Amended) A method according to ~~any of~~ claims 1, 3 or ~~and~~ 5, comprising the step of signaling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison.

Claim 7 (cancelled).

8. (Original) A method according to claim 1, wherein step (C) comprises comparing any of registers, memory, flags, interrupts, tasks, and events in the respective processes.

9. (Original) A method according to claim 1, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

10. (Currently Amended) A control apparatus, comprising

A. any of a first computer process and first computer thread, ~~(collectively, "first process")~~ collectively, "first process, executing a first sequence of instructions,

B. any of a second computer process and second computer thread, ~~(collectively, "second process")~~ collectively, "second process", executing a second sequence of instructions,

C. comparison logic in communication with the first and second processes, the comparison logic comparing a state of the first process to a state of the second process following execution by them of the respective first and second sequences of instructions, the comparison logic signalling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison,

D. scheduling logic in communication with the first and second processes and in communication with the comparison logic, the scheduling logic responding to an error by rolling back each of the first and second processes to prior states in which a favorable comparison was achieved,

E.~~D.~~ wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server, industrial control device, an industrial control system, an environmental control device, an environmental control system, other control device, and other control system.

11. (Original) An apparatus according to claim 10, wherein the comparison logic compares any of registers, memory, flags, interrupts, tasks, and events in each of the respective processes.

12. (Original) An apparatus according to claim 11, wherein the comparison logic comprises

a first comparison logic section operating in the first process for comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence, and

a second comparison logic section operating in the second process for comparing a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence.

13. (Currently Amended) An apparatus according to claim 12, wherein ~~including scheduling logic in communication with the first and second processes and in communication with the comparison logic,~~

the scheduling logic schedules ~~scheduling~~ the first process to execute a third sequence of instructions in response to a favorable comparison by the comparison logic,

the scheduling logic schedules ~~scheduling~~ the second process to execute a fourth sequence of instructions in response to a favorable comparison by the comparison logic.

14. (Original) An apparatus according to claim 13, wherein the comparison logic compares a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

Claims 15 – 16 (cancelled).

17. (Original) An apparatus according to any of claims 10, 13 and 14, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

18. (Original) An apparatus according to claim 10, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

19. (Currently Amended) A process control apparatus, comprising

A. any of a first computer process and a first computer task, ~~(collectively, "first process")~~ collectively, "first process,

B. any of a second computer process and a second computer task, ~~(collectively, "second process")~~ collectively, "second process",

C. scheduling logic in communication with the first and second processes, the scheduling logic scheduling the first process to execute a first subsequence of instructions from a first sequence of instructions, the scheduling logic scheduling the second process to execute a first subsequence of instructions from a second sequence of instructions, the scheduling logic responds to an error by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved,

D. comparison logic in communication with the first and second processes, the comparison logic comparing a state of the first process following completion by it of execution of the first instruction subsequence of instructions with a state of the second process following completion, if any, by it of the second subsequence of instruction, wherein states of the first and second processes include any of registers, memory, flags, interrupts, tasks, and events in those respective processes,

E. synchronization logic in communication with the comparison logic, the synchronization logic responding to a favorable comparison by the comparison logic by scheduling the first process to execute a second subsequence of instructions from the first sequence of instructions and scheduling the second process to execute a second subsequence of instructions from the second sequence of instructions,

20. (Original) An apparatus according to claim 19, wherein the scheduling logic comprises

a first scheduling logic section operating in the first process, the first scheduling logic section scheduling the first process to execute a first subsequence of instructions from a first sequence of instructions, and the first scheduling logic section responding to a favorable comparison by the comparison logic by scheduling the first process to execute a second subsequence of instructions from the first sequence of instructions, and

a second scheduling logic section operating in the second process, the second scheduling logic section scheduling the second process to execute a second subsequence of instructions from a first sequence of instructions, and the second scheduling logic section responding to a favorable comparison by the comparison logic by scheduling the second process to execute a second subsequence of instructions from the second sequence of instructions.

21. (Original) An apparatus according to claim 20, wherein the synchronization logic comprises

a first comparison logic section operating in the first process, the first comparison logic section comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion, if any, by it of the second instruction sequence, and

a second comparison logic section operating in the second process, the second comparison logic section comparing a state of the second process following completion by it of execu-

tion of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence.

22. (Original) An apparatus according to claim 21, wherein at least one of the first and second comparison logic sections signals an error in response to a failure of the first and second processes to achieve comparable states following completion of execution of the respective subsequence of instructions by the respective process.

23. (Original) An apparatus according to claim 22, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

24. (Original) An apparatus according to claim 22, wherein the scheduling logic responds to an error by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

25. (Original) An apparatus according to claim 19, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

26. (Original) An apparatus according to claim 19, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

27. (Currently Amended) A process control apparatus, comprising

A. any of a first computer process and a first computer task, ~~(collectively, "first process")~~ collectively, "first process, executing a first sequence of instructions,

B. any of a second computer process and a second computer task, ~~(collectively, "second process")~~ collectively, "second process", executing a second sequence of instructions,

C. synchronization logic in communication with the first and second processes, the synchronization logic preventing the first process from executing a third sequence of instructions until the first and second processes have completed execution of the first and second sequences of instructions, respectively,

D. at least one of the first and second comparison logic sections signals an error in response to a failure of the first and second processes to achieve comparable states following completion of execution of the respective subsequence of instructions by the respective process. and

E. the scheduling logic responds to an error by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

28. (Original) An apparatus according to claim 27, wherein the synchronization logic prevents the second process from executing a fourth sequence of instructions until the first and second processes have completed execution of the first and second sequences of instructions, respectively.

29. (Original) An apparatus according to any of claims 27 and 28, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

30. (Original) An apparatus according to claim 27, wherein the synchronization logic comprises

a first comparison logic section operating in the first process, the first comparison logic section comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion, if any, by it of the second instruction sequence,

a second comparison logic section operating in the second process, the second comparison logic section comparing a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence,

wherein states of the first and second processes include any of registers, memory, flags, interrupts, tasks, and events in those respective processes.

Claims 31 –32 (cancelled).

33. (Original) An apparatus according to claim 27, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

34. (Original) An apparatus according to claim 27, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

35. (Currently Amended) A method of process control, comprising

A. executing a first sequence of instructions in any of a first computer process and a first computer task ~~(collectively, "first process")~~, collectively, "first process,"

B. executing a second sequence of instructions in any of a second computer process and a second computer task, ~~(collectively, "second process")~~, collectively, "second process," the second process being loosely coupled with respect to the first process,

C. comparing, within the first process, a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion, if any, by it of the second instruction sequence,

D. comparing, within the second process, a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence,

E. wherein any of steps (C) and (D) include the step of signaling an error in response to a failure of the first and second processes to achieve comparable states following completion of execution of the respective instruction sequence by the respective process,

F. responding to such an error by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

36. (Original) A method according to claim 35, including the steps of

scheduling the first process to execute a third sequence of instructions in response to a favorable comparison in step (C), and

scheduling the second process to execute a fourth sequence of instructions in response to a favorable comparison in step (D).

Claim 37 (cancelled).

38. (Original) A method according to claim 37, wherein any of steps (C) and (D) include comparing a state of the respective process with a state of the other process a plurality of times to determine whether they achieve comparable states.

39. (Original) A method according to claim 35, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

40. (Original) A method according to claim 35, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

41. (Currently Amended) A method for process control, comprising

A. executing a first subsequence of instructions from a first sequence of instructions in any of a first computer process and a first computer task, collectively, "first process," ~~(collectively, "first process")~~,

B. executing a first subsequence of instructions from a second sequence of instructions in any of a second computer process and a second computer task, collectively, "second process," ~~(collectively, "second process")~~.

C. comparing a state of the first process following completion by it of execution of the first subsequence of instructions from the first sequence of instructions with a state of the second process following completion, if any, by it of the first subsequence of instructions from the second sequence of instructions,

D. responding to a favorable comparison in step (C) by scheduling the first process to execute a second subsequence of instructions from the first sequence of instructions and scheduling the second process to execute a second subsequence of instructions from the second sequence of instructions,

E. responding to an unfavorable comparison in step (C) by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

42. (Original) A method according to claim 41, wherein step (C) includes

comparing, from within the first process, a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion, if any, by it of the second instruction sequence, and

comparing, from within the second process, a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence.

Claim 43 (cancelled).

44. (Original) A method according to claim 43, wherein step (C) includes comparing a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

45. (Original) A method according to claim 41, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

46. (Original) A method according to claim 41, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

47. (Currently Amended) A method of operation of a digital data processing system, comprising

A. executing a first sequence of instructions in any of a first computer process and first computer thread, ~~(collectively, "first process")~~ collectively, "first process",

B. executing a second sequence of instructions in any of a second computer process and second computer thread, ~~(collectively, "second process")~~collectively, "second process", the second process being loosely coupled with respect to the first process,

C. comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence,

D. responding to failure of the first and second processes to achieve comparable states by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved

48. (Original) A method according to claim 47, comprising the step of executing step (C) one or more times over a time interval in order to determine whether the first and second processes achieve comparable states following completion of execution of the first instruction sequence by the first process.

49. (Original) A method according to claim 48, comprising the step responding to a favorable comparison in step (C) by repeating steps (A) (C) with a third instruction sequence in place of the first instruction sequence, and with a fourth instruction sequence in place of the second instruction sequence.

50. (Original) A method according to claim 49, comprising the steps of

selecting the third instruction sequence as a function of a state of the first process following execution of the first instruction sequence, and

selecting the fourth instruction sequence as a function of a state of the second process following execution of the second instruction sequence.

51. (Original) A method according to claim 50, comprising comparing a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

52. (Currently Amended) A method according to ~~any of~~ claims 47, 49 or ~~and~~ 51, comprising the step of signaling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison.

Claim 53 (cancelled).

54. (Original) A method according to claim 47, wherein step (C) comprises comparing any of registers, memory, flags, interrupts, tasks, and events in the respective processes.

55. (Original) A method according to claim 47, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

56. (Original) A method according to claim 47, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

57. (Currently Amended) A digital data processing apparatus, comprising

A. any of a first computer process and first computer thread, ~~(collectively, "first process")~~ collectively, "first process," executing a first sequence of instructions,

B. any of a second computer process and second computer thread, ~~(collectively, "second process")~~ collectively, "second process", executing a second sequence of instructions,

C. comparison logic in communication with the first and second processes, the comparison logic comparing a state of the first process to a state of the second process following execution by them of the respective first and second sequences of instructions the comparison logic signalling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison,

D. scheduling logic in communication with the first and second processes and in communication with the comparison logic, the scheduling logic responding to an error by rolling back each of the first and second processes to prior states in which a favorable comparison was achieved.

58. (Original) An apparatus according to claim 57, wherein the comparison logic compares any of registers, memory, flags, interrupts, tasks, and events in each of the respective processes.

59. (Original) An apparatus according to claim 58, wherein the comparison logic comprises

a first comparison logic section operating in the first process for comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence, and

a second comparison logic section operating in the second process for comparing a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence.

60. (Original) An apparatus according to claim 59, including scheduling logic in communication with the first and second processes and in communication with the comparison logic,

the scheduling logic scheduling the first process to execute a third sequence of instructions in response to a favorable comparison by the comparison logic,

the scheduling logic scheduling the second process to execute a fourth sequence of instructions in response to a favorable comparison by the comparison logic.

61. (Original) An apparatus according to claim 60, wherein the comparison logic compares a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

Claims 62 –63 (cancelled).

64. (Original) An apparatus according to any of claims 57, 60 and 63, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

65. (Original) An apparatus according to claim 57, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

66. (Original) An apparatus according to claim 57, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

Amendments to the Claims:

This listing of claim will replace all prior versions, and listings, of claims in the application:

1. (Currently Amended) A method of operation of a control system, comprising

A. executing a first sequence of instructions in any of a first computer process and first computer thread, collectively, "first process",

B. executing a second sequence of instructions in any of a second computer process and second computer thread, collectively, "second process", the second process being loosely coupled with respect to the first process,

C. ~~comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence~~ in the first process, receiving a state of the second process and, following execution of the first instruction sequence, comparing a state of the first process with the state of the second process, and

in the second process, receiving a state of the first process and, following execution of the second instruction sequence, comparing a state of the second process with the state of the first process,

D. responding to failure of the first and second processes to achieve comparable states by rolling back each of the first and second processes to prior states in which a favorable comparison indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved, and

E. wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, a process control enterprise server, an industrial control device, an industrial control

system, an environmental control device, an environmental control system, other control device, and other control system.

2. (Original) A method according to claim 1, comprising the step of executing step (C) one or more times over a time interval in order to determine whether the first and second processes achieve comparable states following completion of execution of the first instruction sequence by the first process.

3. (Original) A method according to claim 2, comprising the step responding to a favorable comparison in step (C) by repeating steps (A) - (C) with a third instruction sequence in place of the first instruction sequence, and with a fourth instruction sequence in place of the second instruction sequence.

4. (Original) A method according to claim 3, comprising the steps of

selecting the third instruction sequence as a function of a state of the first process following execution of the first instruction sequence, and

selecting the fourth instruction sequence as a function of a state of the second process following execution of the second instruction sequence.

5. (Original) A method according to claim 4, comprising comparing a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

6. (Previously Presented) A method according to claims 1, 3 or 5, comprising the step of signaling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison.

7. (Cancelled)

8. (Original) A method according to claim 1, wherein step (C) comprises comparing any of registers, memory, flags, interrupts, tasks, and events in the respective processes.

9. (Original) A method according to claim 1, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

10. (Currently Amended) A control apparatus, comprising

A. any of a first computer process and first computer thread, collectively, "first process", executing a first sequence of instructions,

B. any of a second computer process and second computer thread, collectively, "second process", executing a second sequence of instructions,

C. comparison logic in communication with the first and second processes, the comparison logic comprising ~~comparing a state of the first process to a state of the second process following execution by them of the respective first and second sequences of instructions~~

a first comparison logic section operating in the first process, the first comparison logic section configured to exchange a state of the second process and, following execution of the first instruction sequence, compare a state of the first process with the state of the second process, and

a second comparison logic section operating in the second process, the second comparison logic section configured to exchange a state of the first process and, following execution of the second instruction sequence, compare a state of the second process with the state of the first process,

the comparison logic ~~signalling~~ signaling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison,

D. scheduling logic in communication with the first and second processes and in communication with the comparison logic, the scheduling logic responding to an error by rolling back each of the first and second processes to [[a]] prior states in which a favorable comparison indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved, and

E. wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server, industrial control device, an industrial control system, an environmental control device, an environmental control system, other control device, and other control system.

11. (Original) An apparatus according to claim 10, wherein the comparison logic compares any of registers, memory, flags, interrupts, tasks, and events in each of the respective processes.

12. (Cancelled)

13. (Currently Amended) An apparatus according to claim [[12]] 11, wherein,

the scheduling logic schedules the first process to execute a third sequence of instructions in response to a favorable comparison by the comparison logic,

the scheduling logic schedules the second process to execute a fourth sequence of instructions in response to a favorable comparison by the comparison logic.

14. (Original) An apparatus according to claim 13, wherein the comparison logic compares a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

15. - 16. (Cancelled)

17. (Currently Amended) An apparatus according to ~~any of~~ claims 10, 13 [[and]] or 14, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

18. (Original) An apparatus according to claim 10, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

19. (Currently Amended) A process control apparatus, comprising

A. any of a first computer process and a first computer task, collectively, "first process",

B. any of a second computer process and a second computer task, collectively, "second process",

C. scheduling logic in communication with the first and second processes, the scheduling logic scheduling the first process to execute a first subsequence of instructions from a first sequence of instructions, the scheduling logic scheduling the second process to execute a first subsequence of instructions from a second sequence of instructions, the scheduling logic ~~responds~~ responding to an error by rolling back each of the first and second processes to [[a]] prior states in which a favorable comparison indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved,

D. comparison logic in communication with the first and second processes, the comparison logic comprising ~~comparing a state of the first process following completion by it of execution of the first instruction subsequence of instructions with a state of the second process following completion, if any, by it of the second subsequence of instruction,~~

a first comparison logic section operating in the first process, the first comparison logic section configured to exchange a state of the second process and, following execution of the first instruction sequence, compare a state of the first process with the state of the second process, and

a second comparison logic section operating in the second process, the second comparison logic section configured to exchange a state of the first process and, following execution of the second instruction sequence, compare a state of the second process with the state of the first process,

wherein states of the first and second processes include any of registers, memory, flags, interrupts, tasks, and events in those respective processes,

E. synchronization logic in communication with the comparison logic, the synchronization logic responding to a favorable comparison, indicating the first process and the second process executed their respective instruction sequences substantially identically, by the comparison logic by scheduling the first process to execute a second subsequence of instructions from the first sequence of instructions and scheduling the second process to execute a second subsequence of instructions from the second sequence of instructions,

20. (Original) An apparatus according to claim 19, wherein the scheduling logic comprises

a first scheduling logic section operating in the first process, the first scheduling logic section scheduling the first process to execute a first subsequence of instructions from a first sequence of instructions, and the first scheduling logic section responding to a favorable comparison by the comparison logic by scheduling the first process to execute a second subsequence of instructions from the first sequence of instructions, and

a second scheduling logic section operating in the second process, the second scheduling logic section scheduling the second process to execute a second subsequence of instructions from a first sequence of instructions, and the second scheduling logic section responding to a favorable comparison by the comparison logic by scheduling the second process to execute a second subsequence of instructions from the second sequence of instructions.

21. (Cancelled)

22. (Currently Amended) An apparatus according to claim [[21]] 20, wherein at least one of the first and second comparison logic sections signals an error in response to a failure of the first and second processes to achieve comparable states following completion of execution of the respective subsequence of instructions by the respective process.

23. (Original) An apparatus according to claim 22, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

24. (Original) An apparatus according to claim 22, wherein the scheduling logic responds to an error by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

25. (Original) An apparatus according to claim 19, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

26. (Original) An apparatus according to claim 19, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

27. (Currently Amended) A process control apparatus, comprising

A. any of a first computer process and a first computer task, collectively, "first process", executing a first sequence of instructions,

B. any of a second computer process and a second computer task, collectively, "second process", executing a second sequence of instructions,

C. synchronization logic in communication with the first and second processes, the synchronization logic preventing the first process from executing a third sequence of instructions until the first and second processes have completed execution of the first and second sequences of instructions, respectively,

D. a first comparison logic section operating in the first process, the first comparison logic section configured to receive a state of the second process and, following execution of the first instruction sequence, compare a state of the first process with the state of the second process,

a second comparison logic section operating in the second process, the second comparison logic section configured to receive a state of the first process and, following execution of the second instruction sequence, compare a state of the second process with the state of the first process,

at least one of the first and second comparison logic sections signals an error in response to a failure of the first and second processes to achieve comparable states following completion of execution of the respective subsequence of instructions by the respective process, and

E. [[the]] scheduling logic ~~responds~~ configured to respond to an error by rolling back each of the first and second processes to a prior states in which a favorable

comparison indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved.

28. (Original) An apparatus according to claim 27, wherein the synchronization logic prevents the second process from executing a fourth sequence of instructions until the first and second processes have completed execution of the first and second sequences of instructions, respectively.

29. (Currently Amended) An apparatus according to ~~any of~~ claims 27 or ~~and~~ 28, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

30. – 32. (Cancelled)

33. (Original) An apparatus according to claim 27, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

34. (Original) An apparatus according to claim 27, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

35. (Currently Amended) A method of process control, comprising

A. executing a first sequence of instructions in any of a first computer process and a first computer task, collectively, "first process[[,]]",

B. executing a second sequence of instructions in any of a second computer process and a second computer task, collectively, "second process[[,]]", the second process being loosely coupled with respect to the first process,

C. ~~comparing, within the first process, a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion, if any, by it of the second instruction sequence,~~
in the first process, receiving a state of the second process and, following execution of the first instruction sequence, comparing a state of the first process with the state of the second process,

D. ~~comparing, within the second process, a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence.~~
in the second process, receiving a state of the first process and, following execution of the second instruction sequence, comparing a state of the second process with the state of the first process,

E. wherein any of steps (C) and (D) include the step of signaling an error in response to a failure of the first and second processes to achieve comparable states following completion of execution of the respective subsequence of instructions by the respective process, and

F. responding to such an error by rolling back each of the first and second processes to a prior states in which a favorable comparison indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved.

36. (Original) A method according to claim 35, including the steps of
scheduling the first process to execute a third sequence of instructions in response to a favorable comparison in step (C), and

scheduling the second process to execute a fourth sequence of instructions in response to a favorable comparison in step (D).

37. (Cancelled)

38. (Currently Amended) A method according to claim [[37]] 35, wherein any of steps (C) and (D) include comparing a state of the respective process with a state of the other process a plurality of times to determine whether they achieve comparable states.

39. (Original) A method according to claim 35, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

40. (Original) A method according to claim 35, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

41. (Currently Amended) A method for process control, comprising

A. executing a first subsequence of instructions from a first sequence of instructions in any of a first computer process and a first computer task, collectively, "first process[[,]]",

B. executing a first subsequence of instructions from a second sequence of instructions in any of a second computer process and a second computer task, collectively, "second process[[,]]",

C. ~~comparing a state of the first process following completion by it of execution of the first subsequence of instructions from the first sequence of instructions with a state of the second process following completion, if any, by it of the first subsequence of instructions from the second sequence of instructions,~~
in the first process, receiving a state of the second process and, following execution of

the first subsequence of instructions from the first sequence of instructions, comparing a state of the first process with the state of the second process,

in the second process, receiving a state of the first process and, following execution of the first subsequence of instructions from the second sequence of instructions, comparing a state of the second process with the state of the first process,

D. responding to a favorable comparison indicating the first process and the second process executed their respective instruction sequences substantially identically in step (C) by scheduling the first process to execute a second subsequence of instructions from the first sequence of instructions and scheduling the second process to execute a second subsequence of instructions from the second sequence of instructions,

E. responding to an error ~~an unfavorable comparison~~ in step (C) by rolling back each of the first and second processes to [[a]] prior states in which a favorable comparison indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved.

42. – 43. (Cancelled)

44. (Currently Amended) A method according to claim [[43]] 41, wherein step (C) includes comparing a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

45. (Original) A method according to claim 41, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

46. (Original) A method according to claim 41, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

47. (Currently Amended) A method of operation of a digital data processing system, comprising

A. executing a first sequence of instructions in any of a first computer process and first computer thread, collectively, "first process",

B. executing a second sequence of instructions in any of a second computer process and second computer thread, collectively, "second process", the second process being loosely coupled with respect to the first process,

C. ~~comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence.~~

in the first process, exchanging a state of the second process and, following execution of the first instruction sequence, comparing a state of the first process with the state of the second process,

in the second process, exchanging a state of the first process and, following execution of the second instruction sequence, comparing a state of the second process with the state of the first process, and

D. responding to failure of the first and second processes to achieve comparable states by rolling back each of the first and second processes to [[a]] prior states in which a

favorable comparison indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved.

48. (Original) A method according to claim 47, comprising the step of executing step (C) one or more times over a time interval in order to determine whether the first and second processes achieve comparable states following completion of execution of the first instruction sequence by the first process.

49. (Original) A method according to claim 48, comprising the step responding to a favorable comparison in step (C) by repeating steps (A) - (C) with a third instruction sequence in place of the first instruction sequence, and with a fourth instruction sequence in place of the second instruction sequence.

50. (Original) A method according to claim 49, comprising the steps of

selecting the third instruction sequence as a function of a state of the first process following execution of the first instruction sequence, and

selecting the fourth instruction sequence as a function of a state of the second process following execution of the second instruction sequence.

51. (Original) A method according to claim 50, comprising comparing a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

52. (Previously Presented) A method according to claims 47, 49 or 51, comprising the step of signaling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison.

53. (Cancelled)

54. (Original) A method according to claim 47, wherein step (C) comprises comparing any of registers, memory, flags, interrupts, tasks, and events in the respective processes.

55. (Original) A method according to claim 47, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

56. (Original) A method according to claim 47, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

57. (Currently Amended) A digital data processing apparatus, comprising

A. any of a first computer process and first computer thread, collectively, "first process[[,]]", executing a first sequence of instructions,

B. any of a second computer process and second computer thread, collectively, "second process", executing a second sequence of instructions,

C. comparison logic in communication with the first and second processes, the comparison logic comprising ~~comparing a state of the first process to a state of the second process following execution by them of the respective first and second sequences of instructions,~~

a first comparison logic section operating in the first process, the first comparison logic section configured to exchange a state of the second process and, following execution of the first instruction sequence, compare a state of the first process with the state of the second process,

a second comparison logic section operating in the second process, the second comparison logic section configured to exchange a state of the first process and, following execution of the second instruction sequence, compare a state of the second process with the state of the first process.

the comparison logic signaling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison,

D. scheduling logic in communication with the first and second processes and in communication with the first and second comparison logic sections, the scheduling logic responding to an error by rolling back each of the first and second processes to a prior states in which a favorable comparison indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved.

58. (Original) An apparatus according to claim 57, wherein the comparison logic compares any of registers, memory, flags, interrupts, tasks, and events in each of the respective processes.

59. (Cancelled)

60. (Original) An apparatus according to claim 59, including scheduling logic in communication with the first and second processes and in communication with the comparison logic,

the scheduling logic scheduling the first process to execute a third sequence of instructions in response to a favorable comparison by the comparison logic,

the scheduling logic scheduling the second process to execute a fourth sequence of instructions in response to a favorable comparison by the comparison logic.

61. (Original) An apparatus according to claim 60, wherein the comparison logic compares a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

62. – 63. (Cancelled)

64. (Currently Amended) An apparatus according to ~~any of~~ claims 57[[,]] or 60 ~~and 63~~, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

65. (Original) An apparatus according to claim 57, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

66. (Original) An apparatus according to claim 57, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

Proposed Amendments to the Claims:

1. (Proposed Amendment) A computerized method of operation of a control system, comprising

A. with a first scheduler associated with a first process ~~any of a first computer process and first computer thread, collectively, "first process",~~ selecting a highest priority event associated with the first process, where the first process is any of a first computer process and a first computer thread,

B. with a second scheduler associated with a second process ~~any of a second computer process and second computer thread, collectively, "second process",~~ the second process being loosely coupled with respect to the first process, selecting a highest priority event associated with the second process, where the second process is any of a second computer process and a second computer thread,

C. with each of the first scheduler and the second scheduler, comparing for identity of the highest priority event selected by the first scheduler with the highest priority event selected by the second scheduler,

D. with the first scheduler, responding to ~~a favorable comparison~~ an agreement of identity indicated in step (C) by selecting a first sequence of instructions in the first process,

E. with the second scheduler, responding to ~~a favorable comparison~~ an agreement of identity indicated in step (C) by selecting a second sequence of instructions in the second process,

F. with each of the first and second schedulers, comparing for identity of the selection made by the first scheduler with the selection made by the second scheduler,

G. with the first scheduler, responding to ~~a favorable comparison~~ an agreement of identity indicated in step (F) by executing, in the first process, the first sequence of instructions,

H. with the second scheduler, responding to ~~a favorable comparison~~ an agreement of identity indicated in step (F) by executing, in the second process, the second sequence of instructions,

I. responding to ~~an unfavorable comparison~~ a non-agreement of identity occurring more than a selected number of times in any of steps (C) and (F) by rolling back each of the first and second processes to prior states in which ~~a favorable comparison~~ an agreement of identity indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved, and

J. wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, a process control enterprise server, an industrial control device, an industrial control system, an environmental control device, an environmental control system, other control device, and other control system.

2. (Proposed Amendment) A method according to claim 1, comprising the step of executing steps (C) – (H) ~~step (C)~~ one or more times over a time interval in order to determine whether the first and second processes achieve comparable states following completion of execution of the first ~~instruction~~ sequence of instructions by the first process.

3. (Proposed Amendment) A method according to claim 2, comprising the step of responding to ~~a favorable comparison~~ an agreement of identity indicated in ~~step (C)~~ steps (C) and (F) by repeating steps ~~(A) – (C)~~ (A) – (I) with a third instruction sequence in place of the first ~~instruction~~ sequence of instructions, and with a fourth instruction sequence in place of the second ~~instruction~~ sequence of instructions.

4. (Proposed Amendment) A method according to claim 3, comprising the steps of

selecting the third instruction sequence as a function of a state of the first process following execution of the first ~~instruction~~ sequence of instructions, and

selecting the fourth instruction sequence as a function of a state of the second process following execution of the second ~~instruction~~ sequence of instructions.

5. (Proposed Amendment) A method according to claim 4, comprising comparing [[a]] the state of the first process prior to execution by it of the third instruction sequence with [[a]] the state of the second process prior to execution by it of the fourth instruction sequence.

6. (Previously Presented) A method according to claims 1, 3 or 5, comprising the step of signaling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison.

7. (Cancelled)

8. (Proposed Amendment) A method according to claim 1, wherein any of steps ~~step~~ (C) and (F) comprises comparing any of registers, memory, flags, interrupts, tasks, and events in the respective processes.

9. (Original) A method according to claim 1, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

10. (Proposed Amendment) A control apparatus comprising one of more digital data processors capable of executing any of a process or a thread, comprising

A. a first scheduler associated with a first process ~~any of a first computer process and first computer thread, collectively, "first process",~~ the first scheduler configured to select a highest priority event associated with the first process, where the first process is any of a first computer process and a first computer thread,

B. a second scheduler associated with a second process ~~any of a second computer process and second computer thread, collectively, "second process",~~ the second scheduler configured to select a highest priority event associated with the second process, where the second process is any of a second computer process and a second computer thread,

C. each of the first scheduler and the second scheduler further configured to compare for identity of the highest priority event selected by the first scheduler with the highest priority event selected by the second scheduler,

D. the first scheduler further configured to respond to ~~a favorable comparison~~ an agreement of identity indicated in step (C) by selecting a first sequence of instructions in the first process,

E. the second scheduler further configured to respond to ~~a favorable comparison~~ an agreement of identity indicated in step (C) by selecting a second sequence of instructions in the second process,

F. each of the first scheduler and the second scheduler further configured to compare for identity of the selection made by the first scheduler with the selection made by the second scheduler,

G. the first scheduler further configured to respond to ~~a favorable comparison~~ an agreement of identity indicated in step (F) by executing, in the first process, the first sequence of instructions,

H. the second scheduler further configured to respond to ~~a favorable comparison~~ an agreement of identity indicated in step (F), by executing, in the second process, the second sequence of instructions,

I[[D]]. each of the first scheduler and the second scheduler further configured to respond to ~~an unfavorable comparison~~ a non-agreement of identity occurring more than a selected number of times in any of steps (C) and (F) by rolling back each of the first and second processes to prior states in which ~~a favorable comparison~~ an agreement of identity indicating the first process and the second process executed their respective instruction sequences substantially identically was achieved, and

J[[E]]. wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server, industrial control device, an industrial control system, an environmental control device, an environmental control system, other control device, and other control system.

11. (Proposed Amendment) An apparatus according to claim 10, wherein any of the first scheduler and the second scheduler is configured to compare ~~the comparison logic compares~~ any of registers, memory, flags, interrupts, tasks, and events in each of the respective processes.

12. (Cancelled)

13. (Proposed Amendment) An apparatus according to claim 11, wherein,

the first scheduler is configured to ~~scheduling logic schedules the first process to execute~~ select a third sequence of instructions as a function of a state of the first process following execution of the first sequence of instructions ~~in response to a favorable comparison by the comparison logic~~,

the second scheduler is configured to ~~scheduling logic schedules the second process to execute~~ select a fourth sequence of instructions as a function of a state of the second process following execution of the second sequence of instructions ~~in response to a favorable comparison by the comparison logic~~.

14. (Proposed Amendment) An apparatus according to claim 13, wherein each of the first and second schedulers are configured to compare the ~~comparison logic compares a~~ state of the first process prior to execution by it of the third instruction sequence with [[a]] the state of the second process prior to execution by it of the fourth instruction sequence.

15. - 16. (Cancelled)

17. (Proposed Amendment) An apparatus according to claims 10, 13 or 14, wherein ~~the comparison logic compares~~ each of the first and second schedulers are configured to compare a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

18. (Original) An apparatus according to claim 10, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

19. – 66. (Cancelled)

FAX TRANSMISSION

DATE: 2/6/06

PTO IDENTIFIER: Application Number 09/328,828-Conf. #7344
Patent Number
Inventor: Samuel Galpin

MESSAGE TO: Commissioner for Patents

FAX NUMBER: (703) 308-6642

FROM: NUTTER MCCLENNEN & FISH LLP
David J. Powsner

PHONE: (617) 439-2000

Attorney Dkt. #: 102314-0082

PAGES (Including Cover Sheet): 44 (Part 1 of 2)

CONTENTS:
Response to Notice to File Corrected Application Papers (1 page)
Combined Declaration and Power of Attorney (including Exhibit A) (41 pages)
Certificate of Transmission (1 page)

NUTTER MCCLENNEN & FISH LLP
World Trade Center West, 155 Seaport Boulevard, Boston, Massachusetts 02210-2604
Telephone: (617) 439-2000 Facsimile: (617) 310-9000

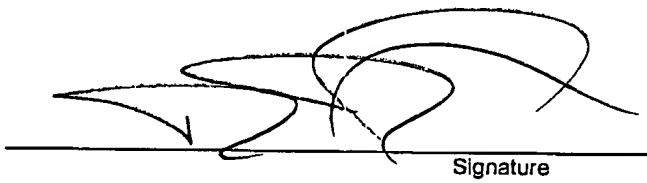
PTO/SB/97 (09-04)
Approved for use through 07/31/2006. OMB 0651-0031
U. S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Application No. (If known): 09/328,828 Attorney Docket No.: 102314-0082

Certificate of Transmission under 37 CFR 1.8

I hereby certify that this correspondence is being facsimile transmitted to the United States Patent and Trademark Office.

on 2/6/06 .
Date



Signature

David J. Powsner
Typed or printed name of person signing Certificate

31,868
Registration Number, if applicable

(617) 439-2717
Telephone Number

Note: Each paper must have its own certificate of transmission, or this certificate must identify each submitted paper.

Response to Notice to File Corrected Application Papers (1 page)
Combined Declaration and Power of Attorney (inculding Exhibit A) (41 pages)

1501591.1

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being transmitted by facsimile to the Patent and Trademark Office, facsimile no. (703) 308-6642, on the date shown below.

Dated: 2/6/06 Signature: _____ (David J. Powsner)

Docket No.: 102314-0082 (PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Samuel Galpin

Application No.: 09/328,828

Filed: June 8, 1999

Confirmation No.: 7344

Art Unit: 2127

For: METHODS AND APPARATUS FOR FAULT-DETECTING AND FAULT-TOLERANT PROCESS CONTROL

Examiner: K. Tang

RESPONSE TO NOTICE TO FILE CORRECTED APPLICATION PAPERS

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the Notice to File Corrected Application Papers mailed January 6, 2006, enclosed are: a Copy of Notice to File Corrected Application Papers, a Combined Declaration and Power of Attorney, an Application Data Sheet, and a Certificate of Facsimile Transmission.

The Director is hereby authorized to charge any deficiency in the fees filed, asserted to be filed or which should have been filed herewith (or with any paper hereafter filed in this application by this firm) to our Deposit Account No. 141449, under Order No. 102314-0082.

Dated: 2/6/06

Respectfully submitted,

By _____
David J. Powsner
Registration No.: 31,868
NUTTER MCCLENNEN & FISH LLP
World Trade Center West
155 Seaport Boulevard
Boston, Massachusetts 02210-2604
(617) 439-2717
(617) 310-9717 (Fax)
Attorney for Applicant

Declaration for Patent Application
I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the U.S. Postal Service on the date shown below with sufficient postage as First Class Mail, in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
Dated: 2/6/06 Signature: _____
(David J. Powsner)

Attorney Docket No.: 102314-0082

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

DECLARATION FOR PATENT APPLICATION

As the below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHODS AND APPARATUS FOR FAULT-DETECTING AND FAULT-TOLERANT PROCESS CONTROL

the specification of which was filed on June 8, 1999 as Application No. 09/328,828 and is attached hereto as Exhibit A.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims as originally filed, and as amended by an amendment, if any, specifically referred to in the original filing.

I hereby also declare that the subject matter of the amendments filed on November 22, 2004, July 5, 2005, and September 9, 2005, all attached hereto as Exhibit B, was part of the invention and was invented before the filing date of the original application, above identified for such invention.

I acknowledge the duty to disclose all information known to me that is material to patentability in accordance with Title 37, Code of Federal Regulations, § 1.56.

FOREIGN PRIORITY CLAIM

I hereby claim foreign priority benefits under Title 35, United States Code § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

[x] no such foreign applications have been filed

[ ] such foreign application have been filed as follows:

Attorney Docket No.: 102314-0082

EARLIEST FOREIGN APPLICATION(S), IF ANY FILED WITHIN 12 MONTHS (6 MONTHS FOR DESIGN) PRIOR TO THIS U.S. APPLICATION

Application Number	Country	Date of Filing	Priority Claimed Under 35 USC 119
			___ Yes No ___
			___ Yes No ___
			___ Yes No ___

ALL FOREIGN APPLICATION(S), IF ANY FILED MORE THAN 12 MONTHS (6 MONTHS FOR DESIGN) PRIOR TO THIS U.S. APPLICATION

Application Number	Country	Date of Filing

CLAIM FOR BENEFIT OF EARLIER U.S. PROVISIONAL APPLICATIONS

I hereby claim priority benefits under Title 35, United States Code §119(e), of any United States provisional patent application(s) listed below:

[x] no such U.S. provisional applications have been filed.

[ ] such U.S. provisional application have been filed as follows:

Application Number	Date of Filing	Priority Claimed Under 35 USC 119
		___ Yes No ___
		___ Yes No ___
		___ Yes No ___

CLAIM FOR BENEFIT OF EARLIER U.S./PCT APPLICATION(S)

I hereby claim the benefit under Title 35, United States Code, §120 of the United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose

all information that is material to patentability in accordance with Title 37, Code of Federal Regulations, §1.56 which became available to me between the filing date of the prior application and the national or PCT international filing date of this application:

[x] no such U.S./PCT applications have been filed.

[ ] such U.S./PCT application have been filed as follows:

Application Number	Date of Filing	Status (Patented/Pending/Abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

I hereby appoint:

All practitioners at Customer Number 021125

all of **Nutter McClennen & Fish LLP**, World Trade Center West, 155 Seaport Boulevard, Boston, Massachusetts 02210-2604, jointly, and each of them severally, my attorneys at law/patent agent(s), with full power of substitution, delegation and revocation, to prosecute this application, to make alterations and amendments therein, to receive the patent, and to transact all business in the U. S. Patent and Trademark Office connected therewith.

Please mail all correspondence to David J. Powsner at **Customer Number** 021125, whose address is:

Nutter McClennen & Fish LLP
World Trade Center West
155 Seaport Boulevard
Boston, Massachusetts 02210-2604

Please direct telephone calls to: David J. Powsner at (617) 439-2000.

Please direct facsimiles to: (617) 310-9717

Attorney Docket No.: 102314-0082

Full name of sole or first inventor Samuel Galpin	
Sole or first inventor's signature	Date JAN 25, 2006
Residence Hingham, Massachusetts	
Citizenship US	
Mailing Address 38 Highview Drive Hingham, Massachusetts 02043	

1496124.1

Exhibit A

U.S. Postal Service Express Mail Label No. EE776165370US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION FOR

METHODS AND APPARATUS FOR FAULT-DETECTING AND FAULT-TOLERANT PROCESS CONTROL

Inventor:
Samuel Galpin
a citizen of the United States residing at
38 Highview Drive
Hingham, MA 02043

EE776165370US

METHODS AND APPARATUS FOR FAULT-DETECTING AND FAULT-TOLERANT PROCESS CONTROL

Background of the Invention

The invention pertains to control systems and, more particularly, by way of non-limiting example, to fault-detecting and fault-tolerant methods and apparatus for process control.

The terms "control" and "control systems" refer to the control of the operational parameters of a device or system by monitoring one or more of its characteristics. This is used to insure that output, processing, quality and/or efficiency remain within desired parameters over the course of time.

Control is used number of fields. Process control, for example, is typically employed in the manufacturing sector for process, repetitive and discrete manufactures, though, it also has wide application in electric and other service industries. Environmental control finds application in residential, commercial, institutional and industrial settings, where temperature and other environmental factors must be properly maintained. Control is also used in articles of manufacture, from toasters to aircraft, to monitor and control device operation.

Control systems typically utilize field devices that are physically integrated into the equipment being controlled. For example, temperature sensors are usually installed directly on or within the articles, bins, or conduits that process, contain or transport the materials being measured. Control devices such as valves, relays, and the like, must also be integrated with the equipment whose operations they govern.

Predictability is among the key requirements of any control device. A fluid sensor that even occasionally produces flaky readings is unacceptable. Overengineering can insure better reliability; however, it often results in devices that are too expensive or too large for wide application.

Redundancy is a well accepted alternative to overengineering. It typically involves using two or more standard control elements in place of one. The duplicated units can be field modules, controllers or other higher-level elements in the control hierarchy.

Thus, for example, U.S. Patent 4,347,563 discloses an industrial control system in which redundant processing units serve as bus masters "of the moment," monitoring status information generated by primary processing units. If a redundant unit detects that a primary has gone faulty while executing an applications program, the redundant unit loads that program and takes over the primary's function.

U.S. Patent 5,008,805, on the other hand, discloses a real time control system in which "sender" and "listener" processors synchronously step through sequential schedules, with the sender controlling execution of events sent from a host. The listener monitors the sender and in the event of fault, assumes the role of the latter, executing commands omitted during the takeover interval.

A shortcoming of these and many other prior art redundancy schemes is their imposition of undue computational or hardware overhead. U.S. Patents 5,008,805, for example, has the disadvantage of requiring that the sender and listener operate in lock-step, necessitating common timing lines and up-front synchronization procedures.

An object of this invention is to provide improved methods and apparatus for control and more particularly, for example, for fault-tolerance and fault-detection in process control.

A related object of the invention is to provide such methods and apparatus as demand low computation and hardware overhead and thus, for example, that can be implemented on a wide range of control equipment, from field devices to plant and enterprise servers.

A further object of the invention is to provide such methods and apparatus as can be implemented in new control equipment and retrofitted into pre-existing equipment.

Summary of the Invention

The foregoing are among the objects attained by the invention which provides, in one aspect, a method of process, environmental or other control (hereinafter, "control") that includes executing a first sequence of instructions in a first process or thread (collectively, "process") and executing a second sequence of instructions in a second process. The processes are loosely coupled, that is, they do not operate in forced lock-step synchronism with one another and need not, for example, share a common clock. Though the processes can operate on the same processor, typically, they operate on different processors, e.g., within separate elements of a control system. Thus, for example, according to some aspects of the invention, the first and second processes are resident and execute on respective partners in paired field devices (e.g., sensors), block controllers, process controllers, and plant or enterprise servers.

The method further includes comparing states of the first and second processes following their respective completions of the first and second instruction sequences. The comparison can cover registers, memory, flags, interrupts, tasks, and/or events for each of the respective processes. Thus, for example, to determine whether the two processes executed their respective sequences substantially identically -- and, therefore, are ready to begin to execute additional sequences -- one aspect of the invention calls for comparing flags set during execution by the interrupt handlers of each process. Execution of further instruction sequences by either process is delayed, according to related aspects of the invention, pending a favorable comparison of the process states. To this end, one aspect of the invention calls for comparing the states multiple times, e.g., over a predefined time interval. In addition to compensating for non-significant differences in execution speed or event occurrence, employing multiple comparisons insures synchronism, fault-detection, and fault-tolerance, while effectuating loose coupling of the processes.

If the processes achieve comparable states following completion of the first and second sequences, the first process takes up execution of a third sequence, while the second process takes up execution of a fourth sequence. In the event that one of the processes does not complete its respective instruction sequence, or if the process states do not otherwise favorably compare, an error is signaled.

Alternatively, or in addition, the method calls for rolling back the processes to performing error diagnostics and/or rolling back the processes to their most recent respective states of agreement, e.g., the states prior to execution of the first and second sequences, and retrying execution of those sequences.

Further aspects of the invention provide methods as described above in which the instruction sequences are selected for execution based on the process states. As above, the state information can include registers, interrupts, memory, flags and/or events in each of the respective processes, though one aspect of the invention calls for the selection to be made based on the tasks queued for completion in each process (e.g., as a result of interrupts received during execution of the prior sequences). Selections made for the respective processes, e.g., the identity of the aforementioned third and fourth sequences, can be compared between the processes. As above, the comparison can be conducted over a predetermined interval and, in the event of non-agreement, an error can be signaled and/or the processes rolled back to a prior state.

Still further aspects of the invention provide control apparatus operating in accord with the above-described methods. These can comprise devices at any level of the process control hierarchy, from process control field device (e.g., sensors) to block controllers to process controllers to plant and enterprise servers.

Still other aspects of the invention provide general purpose digital data processing apparatus and methods paralleling the control apparatus and methods described above.

These and other aspects of the invention are evident in the drawings and in the description that follows.

Systems according to the invention have a number of advantages over the prior art. Among these are that they permit control devices to provide fault-detection and achieve fault tolerance without special-purpose hardware. Instead, only a minimal facility to exchange state information between the modules is necessary. The invention, as evident below, can be used with either hardware-based error

detection or software comparison or arbitration. The invention can also be embodied in dual-partnered systems, as well as those that are triply or quadruply redundant.

Brief Description of the Drawings

A more complete understanding of the invention may be attained by reference to the drawings, in which:

Figure 1 depicts a pair of process control field devices according to the invention;

Figure 2 depicts a method of operation of a synchronizing-scheduler in the devices of Figure 1;

Figures 3A and 3B depict a method of operation of a synchronizer in the devices of Figure 1; and

Figure 4 depicts a process control system with field devices, control processors and plant or enterprise servers according to the invention.

Detailed Description of the Illustrated Embodiment

Figure 1 depicts a pair of process control devices 10, 12 according to the invention. The illustrated devices are field devices, though the invention may applied to other control devices as well, such as, by non-limiting example, block controllers, process controllers, plant and enterprise servers, and environmental and industrial controls and controllers.

Illustrated devices 10, 12 include processing sections 18, 20 and sensing apparatus -- in this case, by way of non-limiting example, sensors 14, 16 for measuring flow along pipe 17. The sections 18, 20 are constructed and operated in accord with the teachings herein to provide fault detection and fault tolerance. They can also be configured to provide signal conditioning and other functions commonly known in the art. To these ends, illustrated sections 18, 20 can comprise general or special purpose digital data processing apparatus, including central processing units 22, 24 and various storage devices, such as code stores 26, 28 and event stores 30, 32. These storage devices may be embodied in any conventional storage units, such as, by way of non-limiting example, EEPROMS or DRAMS, and they may be implemented in any conventional manner, e.g., arrays, linked lists, b-trees, and so forth. In addition to, or in lieu of such digital data processing apparatus, illustrated sections 18, 20 may comprise analog electronics or other functionality configured for carrying out the tasks described herein.

Control devices 10, 12 are coupled to one another for the exchange of state information of the type described below. In the illustrated embodiment, such exchange takes place over communication pathway 38, which may be connected to the serial, parallel, or other input/output ports of the respective processing sections. Pathway 38 can be implemented via cable, bus, LAN, WAN, Internet, or other direct, indirect or networked medium. Moreover, it may be implemented via conventional cabling or via infrared, microwave or transmission-based medium.

In addition to being coupled to one another, sections 18, 20 can be coupled to other elements of a process control system, such as, block controllers, process controllers, and plant and enterprise servers. In the illustrated embodiment, such coupling is provided via network 40 (e.g., an Ethernet network). As above, this may be implemented via cable, bus, LAN, WAN, Internet, or other direct,

indirect or networked medium, using conventional cabling, infrared, microwave or transmission-based medium. Those skilled in the art will, of course, appreciate that the information exchange described below as taking place over line 38 can, instead or in addition, take place over network 40.

Illustrated stores 30, 32 contain information reflecting a state of each respective process control device 10, 12. In the illustrated embodiment, these comprise event flags generated by interrupt handlers (not shown) executing on the respective central processing units 22, 24. The flags represent external interrupts, such as those received from sensors 14, 16, from other process control devices coupled to network 40, and from other peripheral attached directly to devices 10, 12. The flags also represent internal interrupts, such as those generated by watch-dog and other timers (not shown) internal to the processing sections 18, 20.

The state of control devices 10, 12 may be reflected by other information, as well. This can include, for example, pending interrupts, task and event queues, and so forth. As discussed below, state information used in the illustrated embodiment also includes the identity portions of code sequences, i.e., subsequences or slices, elected for execution by each device 10, 12, which information may be maintained, for example, in registers or other memory areas in the devices.

Code stores 26, 28 contain computer instructions ("code") for execution by processing sections 18, 20, e.g., in response to events listed in event stores 30, 32. The code, which may of the type generally known in the art of process control or which may be otherwise suitable therefore (whether or not currently known), is preferably parceled into units or "slices" 42 - 52. These slices may be, for example, indivisible or "atomic" sequences of code. They may also be divisible sequences sized for efficient execution in view of the synchronization methodologies described herein. As discussed below, a preferred slice represents an increment of work that is small enough to be completed in a time interval consistent with the delay that can be accepted if the thread executing it is preempted.

Illustrated central processing units 22, 24 are of the conventional type known in the art capable of executing instruction sequences within processes, threads and other such instruction execution contexts (collectively, "processes"). In addition to slices 42 - 52 contained in code stores 26, 28, the

CPUs 22, 24 execute code for purposes of identifying events and scheduling slices (i.e., synchronizing-schedulers 54, 60), synchronizing event identification and slice selection (i.e., synchronizers 56 62), and dispatching slices for execution (i.e., dispatchers 58, 64).

In the illustrated embodiment, code executed by the CPUs 22, 24 and, particularly, code executed from stores 26, 28, is broken up into execution slices 42 - 52. An execution slice is an increment of work that is small enough to be completed in a time interval consistent with the delay that can be accepted if the process or thread executing it is preempted. In addition, each slice starts at the logical beginning of an operation, and ends when the operation is completed. At slice completion, the execution is between contexts.

While this approach can be applied to general purpose software, it is particularly suited to applications that have an appropriately "sliceable" structure. Process control software has this characteristic. Exploiting it accomplishes two things. The first is to reduce the "operating system" to a simple synchronizing-scheduler. The second is that the synchronizing-scheduler 54, 60 simplifies attaining fault tolerance. It is this simplification, among other things, which makes it possible to support fault tolerant operation with the available hardware resources.

Each synchronizing-scheduler 54, 60 recognizes and synchronizes all pending events. As discussed above, events typically result from interrupts, e.g., clock interrupts, communication interrupts, etc., that have occured during execution of a prior slice. Some events, indeed, are a natural result of slice execution itself, e.g., events reflecting the availability of data computed by the prior slice. After each synchronizing-scheduler 54, 60 synchronizer its pending events with the other synchronizing-scheduler (or, other synchronizing-schedulers), it selects a next slice for execution, and synchronizes *that* selection. In some embodiments, the identities of the slices selected for execution are not synchronized; rather, merely, the fact that slices have been selected for execution.

Operation of the illustrated synchronizing-schedulers 54, 60 is shown in Figure 2 and described in the pseudo-code fragment that follows:

while true (step 70)

```
                if rollback_cnt > max_rollbacks break; (step 72)
                if more unsynchronized events then (step 74)
                        select highest priority pending event (step 76)
                        invoke synchronizer to confirm that partner selected same event (step 78)
                        if synchronization was successful (step 80)
                                rollback_cnt == 0 (step 79)
                        else
                                increment rollback_cnt (step 81)
                        endif
                        continue
                endif
                select code slice for execution to process event (step 82)
                invoke synchronizer to confirm that partner selected same slice (step 84)
                if synchronization not successful (step 86)
                        increment rollback_cnt (step 81)
                        continue
                endif
                rollback_cnt == 0 (step 87)
                dispatch to execute selected slice (step 88)
        end while
        signal error (step 91)
        perform error handling (step 92)
```

The type and degree of error handling in step 92 depends on the nature of the application. In some embodiments, no error handling at all is performed after the fault is signalled. In other embodiments, the processes are rolled back to state of last agreement. In still other embodiments, a fault diagnostic procedure is performed to isolate the source of error.

As noted above, devices 10, 12 on both sides of the redundant pair have synchronization, or sync, lines 38 running between them that are used to keep their operation synchronized. This enforces loose coupling between the devices 10, 12 and, more particularly, between their respective processing sections 18, 20. One of the devices 10 is configured to be the master, and the other 12, the shadow. The synchronizer operation for the master 10 is shown in Figure 3A and described in the pseudo-code that follows:

```
        write event/slice code on sync out lines (step 102)
        set timeout register
        initialize sync results to false
```

```
do
        decrement timeout register
        read input from partner (step 104)
while shadow agrees = false and timeout != 0 (step 106)

if timeout register != 0 (step 108)
        write "null" code on sync out lines (step 110)
        do
                decrement timeout register
                read input from partner (step 112)
        while shadow agrees = true and timeout != 0 (step 114)
endif
if timeout register != 0 (step 116)
        sync results = true (success) (step 118)
endif
return sync results
```

The shadow 12 executes a similar synchronization routine, which is described below and shown in Figure 3B.

```
set timeout register initialize sync result
to false
do
        decrement timeout register
        read master synch state
while shadow state != master state and timeout != 0
operation code and timeout register != 0
if timeout register != 0
        write "shadow agrees" on synch line
        do
                decrement timeout register
                read input from partner
        while master state != "null" and timeout != 0
endif
if timeout register != 0
        synch results = success
endif
return synch results
```

As noted above, the invention can be applied to process control apparatus other than field devices 10, 12, as well. A process control system in which the invention is employed in a plurality of pairs of process control devices 120, block controllers 122 and plant servers 124 is illustrated in

Figure 4. Although the processing performed by these various apparatus differs in accord with their levels in the system, their processing sections perform scheduling and synchronization in the manner describeds above in order to provide fault-detection and tolerance.

In one embodiment, the invention is employed in coprocessors that serve as communication controllers for a network of field modules. Such coprocessor can be contained, for example, within one or more of the block controllers 122 of the type shown in Figure 4. Software executing on the CPUs of those coprocessors is both multithreaded and fault tolerant. It constitutes a complete stand-alone executable that sits directly on top of the hardware, preferably, without the need for a separate operating system.

The aformentioned embodiment is multithreaded with priority scheduling. It does this without a general purpose context-preserving task switch mechanism. Instead the work is broken up into execution slices as described above. The basis of the slices chosen for the coprocessors is to associate a slice with the work required to send or receive a particular message over network 40. There are two reasons for this. First, the hardware events associated with the network activity are good synchronization markers. Second, when one considers fault tolerant error recovery scenarios, the last completed network message is a good point to restart from.

Within the aforementioned coprocessors, each channel is an independent stream of communications transactions associated with a set of data structures in memory that they share between them. A channel is also an execution thread. Each channel has a unique priority. On the "host" side of the shared memory interface, each channel is associated with a vrtx task of corresponding priority.

The communications channels are identical except for priority and assigned buffer sizes. The number of channels is controlled by the initialization of master control structures contained in the shared memory. They share a single common set of routines that execute three basic types of slices:

1. routines that convert a list of transaction requests into the corresponding request messages;

2. routines that send the messages and wait for the replies;

3. routines that convert the messages into the required lists of responses

The execution state of each channel is recorded in a channel data structure within the shared memory. Once started, execution of the slice continues until it completes. The stored state information identifies the next slice to be executed, if any, for each channel.

Fault tolerance error scenarios include exception cases where a slice in progress cannot be completed. In these cases, it is simply abandoned. Because the channel data that controls scheduling is not updated until the slice completes, the slice will be restarted from its beginning when/if the execution thread is resumed.

At a logical and priority level above the communications channels, the master control structures provide what can be thought of as a master control channel. Associated actions at this level always complete. They are independent slices. They range from bus switching to mechanisms for fault tolerance exception handling. Each synchronizing-scheduler within the coprocessors checks the master control structure, then the channel information in priority order, to identify the highest priority slice available for execution. Interrupts are serviced during slice execution, but any resulting thread switch occurs after the current slice is completed.

After a synchronizing-scheduler selects the next slice, it invokes its respective synchronizer. During married fault tolerant operation, the synchronizers in the master and shadow use hardware handshake lines and logic as described to ensure that the both sides choose the same next slice. If this process fails, the synchronizers proceed as described above.

In this coprocessor embodiment, the sync lines transmit data with the following values and meanings:

0. NULL (no sync state being declared)
1. Entering sched
2. Host interrupt
3. Transmit sync
4. Receive event
5. Time event
6. Interrupt to Host
7. Error case

Continuing discussion of the aforementioned embodiment, the synchronizer must be able to unambiguously establish success or failure and it must always return the same value on both sides. To this end, the hardware handshake in the coprocessor uses three parallel lines from master to shadow and 1 from shadow to master. The logic above has a timing window. If the shadow sets the "shadows agress" true just as the master times out, the master will return failure and the shadow will return success. The master mus to continue to assert the synch code for a brief intercval after it sees the shadow agrees signal. The shadow then checks that the masters synch code is set to null slightly after the agree line goes true. Those skilled in the art will, of course, appreciate that operation of any given synchronizer will depend strongly on the number of different states that require recognition and the hardware resources available to support it.

Described herein are methods and apparatus meeting the desired objects. Those skilled in the art will appreciate that the specific embodiments shown in the drawings and described above are examples of the invention and that other embodiments incorporating changes therein also fall within the scope of the invention. Thus, for example, it will be appreciated that the invention can be applied to process, industrial, environmental and other control systems and devices utilizing triple modular redundancy, quaduple modular redundancy and still higher levels of redundancy, as well as to those utilizing paired units (or double modular redundancy) as shown in the examples herein. By way of further example, it will be appreciated that the partnered processes can exchange and compare state information different than that described here. By way of still further example, it will

be appreciated that the invention can be applied to digital data processing apparatus and systems, as well.

In view of the foregoing what we claim is:

1. A method of operation of a control system, comprising

A. executing a first sequence of instructions in any of a first process and first thread (collectively, "first process"),

B. executing a second sequence of instructions in any of a second process and second thread (collectively, "second process"), the second process being loosely coupled with respect to the first process,

C. comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence, and

D. Wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, a process control enterprise server, an industrial control device, an industrial control system, an environmental control device, an environmental control system, other control device, and other control system.

2. A method according to claim 1, comprising the step of executing step (C) one or more times over a time interval in order to determine whether the first and second processes achieve comparable states following completion of execution of the first instruction sequence by the first process.

3. A method according to claim 2, comprising the step responding to a favorable comparison in step (C) by repeating steps (A) - (C) with a third instruction sequence in place of the first instruction sequence, and with a fourth instruction sequence in place of the second instruction sequence.

4. A method according to claim 3, comprising the steps of

selecting the third instruction sequence as a function of a state of the first process following execution of the first instruction sequence, and

selecting the fourth instruction sequence as a function of a state of the second process following execution of the second instruction sequence.

5. A method according to claim 4, comprising comparing a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

6. A method according to any of claims 1, 3 and 5, comprising the step of signaling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison.

7. A method according to claim 6, comprising the step of rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

8. A method according to claim 1, wherein step (C) comprises comparing any of registers, memory, flags, interrupts, tasks, and events in the respective processes.

9. A method according to claim 1, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

10. A control apparatus, comprising

A. any of a first process and first thread (collectively, "first process") executing a first sequence of instructions,

B. any of a second process and second thread (collectively, "second process") executing a second sequence of instructions,

C. comparison logic in communication with the first and second processes, the comparison logic comparing a state of the first process to a state of the second process following execution by them of the respective first and second sequences of instructions.

D. Wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server, industrial control device, an industrial control system, an environmental control device, an environmental control system, other control device, and other control system.

11. An apparatus according to claim 10, wherein the comparison logic compares any of registers, memory, flags, interrupts, tasks, and events in each of the respective processes.

12. An apparatus according to claim 11, wherein the comparison logic comprises

a first comparison logic section operating in the first process for comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence, and

a second comparison logic section operating in the second process for comparing a state of the second process following completion by it of execution of the second instruction

sequence with a state of the first process following completion, if any, by it of the first instruction sequence.

13. An apparatus according to claim 12, including scheduling logic in communication with the first and second processes and in communication with the comparison logic,

the scheduling logic scheduling the first process to execute a third sequence of instructions in response to a favorable comparison by the comparison logic,

the scheduling logic scheduling the second process to execute a fourth sequence of instructions in response to a favorable comparison by the comparison logic.

14. An apparatus according to claim 13, wherein the comparison logic compares a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

15. An apparatus according to claim 10, wherein the comparison logic signals an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison.

16. An apparatus according to claim 15, wherein the scheduling logic responds to an error by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

17. An apparatus according to any of claims 10, 13 and 14, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

18. An apparatus according to claim 10, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

19. A process control apparatus, comprising

A. any of a first process and a first task (collectively, "first process"),

B. any of a second process and a second task (collectively, "second process"),

C. scheduling logic in communication with the first and second processes, the scheduling logic scheduling the first process to execute a first subsequence of instructions from a first sequence of instructions, the scheduling logic scheduling the second process to execute a first subsequence of instructions from a second sequence of instructions,

D. comparison logic in communication with the first and second processes, the comparison logic comparing a state of the first process following completion by it of execution of the first instruction subsequence of instructions with a state of the second process following completion, if any, by it of the second subsequence of instruction, wherein states of the first and second processes include any of registers, memory, flags, interrupts, tasks, and events in those respective processes,

E. synchronization logic in communication with the comparison logic, the synchronization logic responding to a favorable comparison by the comparison logic by scheduling the first process to execute a second subsequence of instructions from the first sequence of instructions and scheduling the second process to execute a second subsequence of instructions from the second sequence of instructions,

20. An apparatus according to claim 19, wherein the scheduling logic comprises

a first scheduling logic section operating in the first process, the first scheduling logic section scheduling the first process to execute a first subsequence of instructions from a first sequence of instructions, and the first scheduling logic section responding to a favorable comparison by the comparison logic by scheduling the first process to execute a second subsequence of instructions from the first sequence of instructions, and

a second scheduling logic section operating in the second process, the second scheduling logic section scheduling the second process to execute a second subsequence of instructions from a first sequence of instructions, and the second scheduling logic section responding to a favorable comparison by the comparison logic by scheduling the second process to execute a second subsequence of instructions from the second sequence of instructions.

21. An apparatus according to claim 20, wherein the synchronization logic comprises

a first comparison logic section operating in the first process, the first comparison logic section comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion, if any, by it of the second instruction sequence, and

a second comparison logic section operating in the second process, the second comparison logic section comparing a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence.

22. An apparatus according to claim 21, wherein at least one of the first and second comparison logic sections signals an error in response to a failure of the first and second

processes to achieve comparable states following completion of execution of the respective subsequence of instructions by the respective process.

23. An apparatus according to claim 22, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

24. An apparatus according to claim 22, wherein the scheduling logic responds to an error by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

25. An apparatus according to claim 19, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

26. An apparatus according to claim 19, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

27. A process control apparatus, comprising

A. any of a first process and a first task (collectively, "first process") executing a first sequence of instructions,

B. any of a second process and a second task (collectively, "second process") executing a second sequence of instructions,

C. synchronization logic in communication with the first and second processes, the synchronization logic preventing the first process from executing a third sequence of

instructions until the first and second processes have completed execution of the first and second sequences of instructions, respectively.

28. An apparatus according to claim 27, wherein the synchronization logic prevents the second process from executing a fourth sequence of instructions until the first and second processes have completed execution of the first and second sequences of instructions, respectively.

29. An apparatus according to any of claims 27 and 28, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

30. An apparatus according to claim 27, wherein the synchronization logic comprises

a first comparison logic section operating in the first process, the first comparison logic section comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion, if any, by it of the second instruction sequence,

a second comparison logic section operating in the second process, the second comparison logic section comparing a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence,

wherein states of the first and second processes include any of registers, memory, flags, interrupts, tasks, and events in those respective processes.

31. An apparatus according to claim 30, wherein at least one of the first and second comparison logic sections signals an error in response to a failure of the first and second processes to achieve comparable states following completion of execution of the respective subsequence of instructions by the respective process.

32. An apparatus according to claim 31, wherein the scheduling logic responds to an error by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

33. An apparatus according to claim 27, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

34. An apparatus according to claim 27, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

35. A method of process control, comprising

A. executing a first sequence of instructions in any of a first process and a first task (collectively, "first process"),

B. executing a second sequence of instructions in any of a second process and a second task (collectively, "second process"), the second process being loosely coupled with respect to the first process,

C. comparing, within the first process, a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion, if any, by it of the second instruction sequence,

D. comparing, within the second process, a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence.

36. A method according to claim 35, including the steps of

scheduling the first process to execute a third sequence of instructions in response to a favorable comparison in step (C), and

scheduling the second process to execute a fourth sequence of instructions in response to a favorable comparison in step (D).

37. A method according to claim 35, wherein any of steps (C) and (D) include the step of signaling an error in response to a failure of the first and second processes to achieve comparable states following completion of execution of the respective instruction sequence by the respective process.

38. A method according to claim 37, wherein any of steps (C) and (D) include comparing a state of the respective process with a state of the other process a plurality of times to determine whether they achieve comparable states.

39. A method according to claim 35, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

40. A method according to claim 35, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

41. A method for process control, comprising

A. executing a first subsequence of instructions from a first sequence of instructions in any of a first process and a first task (collectively, "first process"),

B. executing a first subsequence of instructions from a second sequence of instructions in any of a second process and a second task (collectively, "second process"),

C. comparing a state of the first process following completion by it of execution of the first subsequence of instructions from the first sequence of instructions with a state of the second process following completion, if any, by it of the first subsequence of instructions from the second sequence of instructions,

D. responding to a favorable comparison in step (C) by scheduling the first process to execute a second subsequence of instructions from the first sequence of instructions and scheduling the second process to execute a second subsequence of instructions from the second sequence of instructions.

42. A method according to claim 41, wherein step (C) includes

comparing, from within the first process, a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion, if any, by it of the second instruction sequence, and

comparing, from within the second process, a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence.

43. A method according to claim 42, wherein step (C) includes responding to a failure of the first and second processes to achieve comparable states following completion of execution of the respective subsequence of instructions by the respective process.

44. A method according to claim 43, wherein step (C) includes comparing a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

45. A method according to claim 41, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

46. A method according to claim 41, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

47. A method of operation of a digital data processing system, comprising

A. executing a first sequence of instructions in any of a first process and first thread (collectively, "first process"),

B. executing a second sequence of instructions in any of a second process and second thread (collectively, "second process"), the second process being loosely coupled with respect to the first process,

C. comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence.

48. A method according to claim 47, comprising the step of executing step (C) one or more times over a time interval in order to determine whether the first and second processes achieve comparable states following completion of execution of the first instruction sequence by the first process.

49. A method according to claim 48, comprising the step responding to a favorable comparison in step (C) by repeating steps (A) - (C) with a third instruction sequence in place of the first instruction sequence, and with a fourth instruction sequence in place of the second instruction sequence.

50. A method according to claim 49, comprising the steps of

selecting the third instruction sequence as a function of a state of the first process following execution of the first instruction sequence, and

selecting the fourth instruction sequence as a function of a state of the second process following execution of the second instruction sequence.

51. A method according to claim 50, comprising comparing a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

52. A method according to any of claims 47, 49 and 51, comprising the step of signaling an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison.

53. A method according to claim 52, comprising the step of rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

54. A method according to claim 47, wherein step (C) comprises comparing any of registers, memory, flags, interrupts, tasks, and events in the respective processes.

55. A method according to claim 47, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

56. A method according to claim 47, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

57. A digital data processing apparatus, comprising

A. any of a first process and first thread (collectively, "first process") executing a first sequence of instructions,

B. any of a second process and second thread (collectively, "second process") executing a second sequence of instructions,

C. comparison logic in communication with the first and second processes, the comparison logic comparing a state of the first process to a state of the second process following execution by them of the respective first and second sequences of instructions.

58. An apparatus according to claim 57, wherein the comparison logic compares any of registers, memory, flags, interrupts, tasks, and events in each of the respective processes.

59. An apparatus according to claim 58, wherein the comparison logic comprises

a first comparison logic section operating in the first process for comparing a state of the first process following completion by it of execution of the first instruction sequence with a state of the second process following completion by it of the second instruction sequence, and

a second comparison logic section operating in the second process for comparing a state of the second process following completion by it of execution of the second instruction sequence with a state of the first process following completion, if any, by it of the first instruction sequence.

60. An apparatus according to claim 59, including scheduling logic in communication with the first and second processes and in communication with the comparison logic,

the scheduling logic scheduling the first process to execute a third sequence of instructions in response to a favorable comparison by the comparison logic,

the scheduling logic scheduling the second process to execute a fourth sequence of instructions in response to a favorable comparison by the comparison logic.

61. An apparatus according to claim 60, wherein the comparison logic compares a state of the first process prior to execution by it of the third instruction sequence with a state of the second process prior to execution by it of the fourth instruction sequence.

62. An apparatus according to claim 57, wherein the comparison logic signals an error in response to a failure of the first and second processes to achieve comparable states at a time of comparison.

63. An apparatus according to claim 62, wherein the scheduling logic responds to an error by rolling back each of the first and second processes to a prior states in which a favorable comparison was achieved.

64. An apparatus according to any of claims 57, 60 and 63, wherein the comparison logic compares a state of the first process with a state of the second process a plurality of times to determine whether the first and second processes to achieve comparable states.

65. An apparatus according to claim 57, wherein each of the first and second processes comprise any of a thread and a process, and wherein the first and second processes execute on any of the same and different digital data processing apparatus.

66. An apparatus according to claim 57, comprising wherein each of the first and second processes execute on any of a process control field device, a block controller, a process controller, a process control plant server, and a process control enterprise server.

Abstract

A method of process, industrial, environmental or other control includes executing a first sequence of instructions in a first process (or thread) and executing a second sequence of instructions in a second process (or thread) that is loosely coupled with the first. States of the first and second processes are compared following their respective completions of the first and second instruction sequences. The comparison can cover registers, memory, flags, interrupts, tasks, and/or events in each of the respective processes. Execution of further instruction sequences by either process is delayed pending a favorable comparison of the process states. If the process achieve comparable states, the first process can take up execution of a third sequence, while the second process takes up execution of a fourth sequence. In the event that one of the processes does not complete its respective instruction sequence within a set period of time, or if the process states do not otherwise favorably compare after execution of the respective sequences, the method calls for signaling an error. A device for process, industrial, environmental or other control operates in accord with such a method.

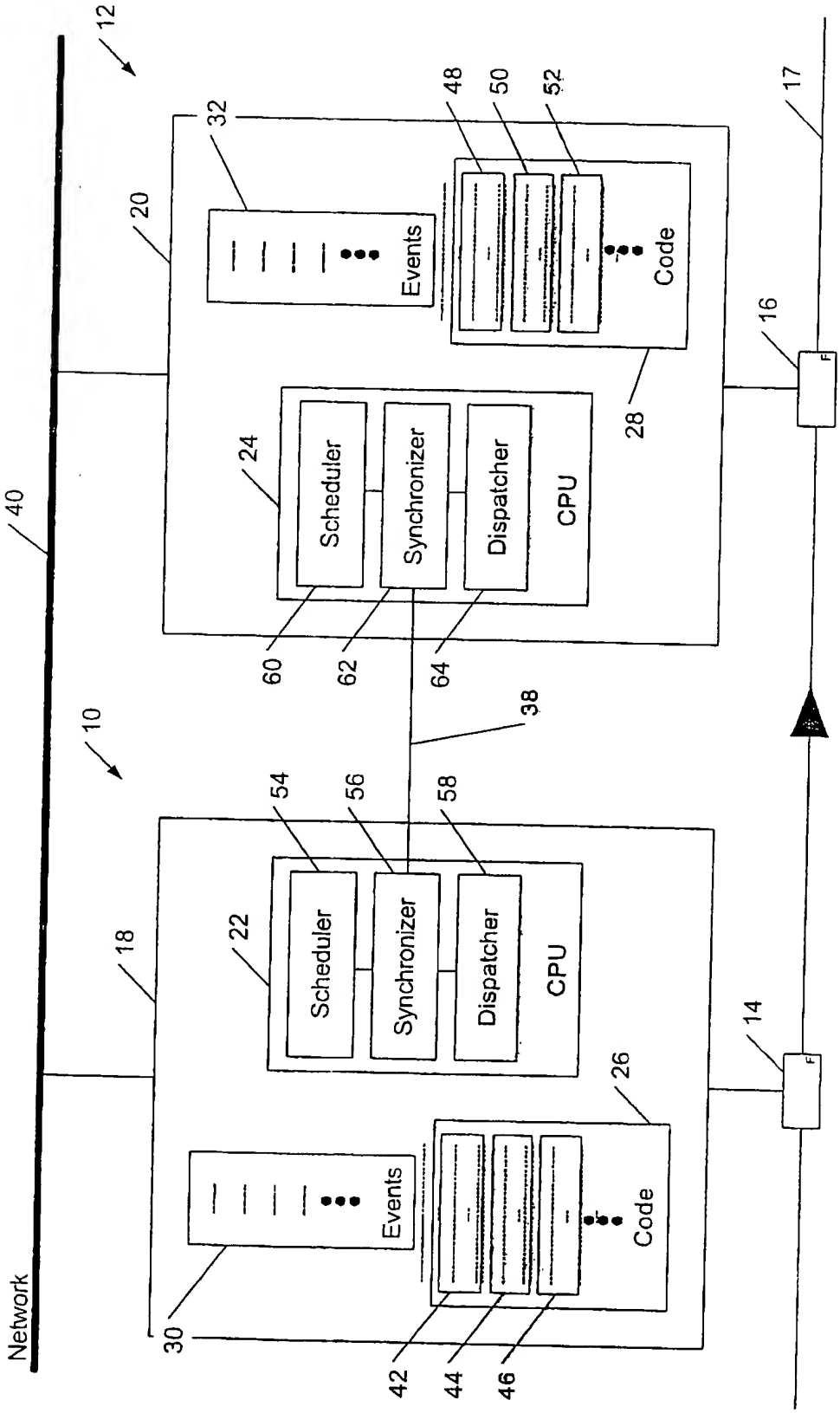
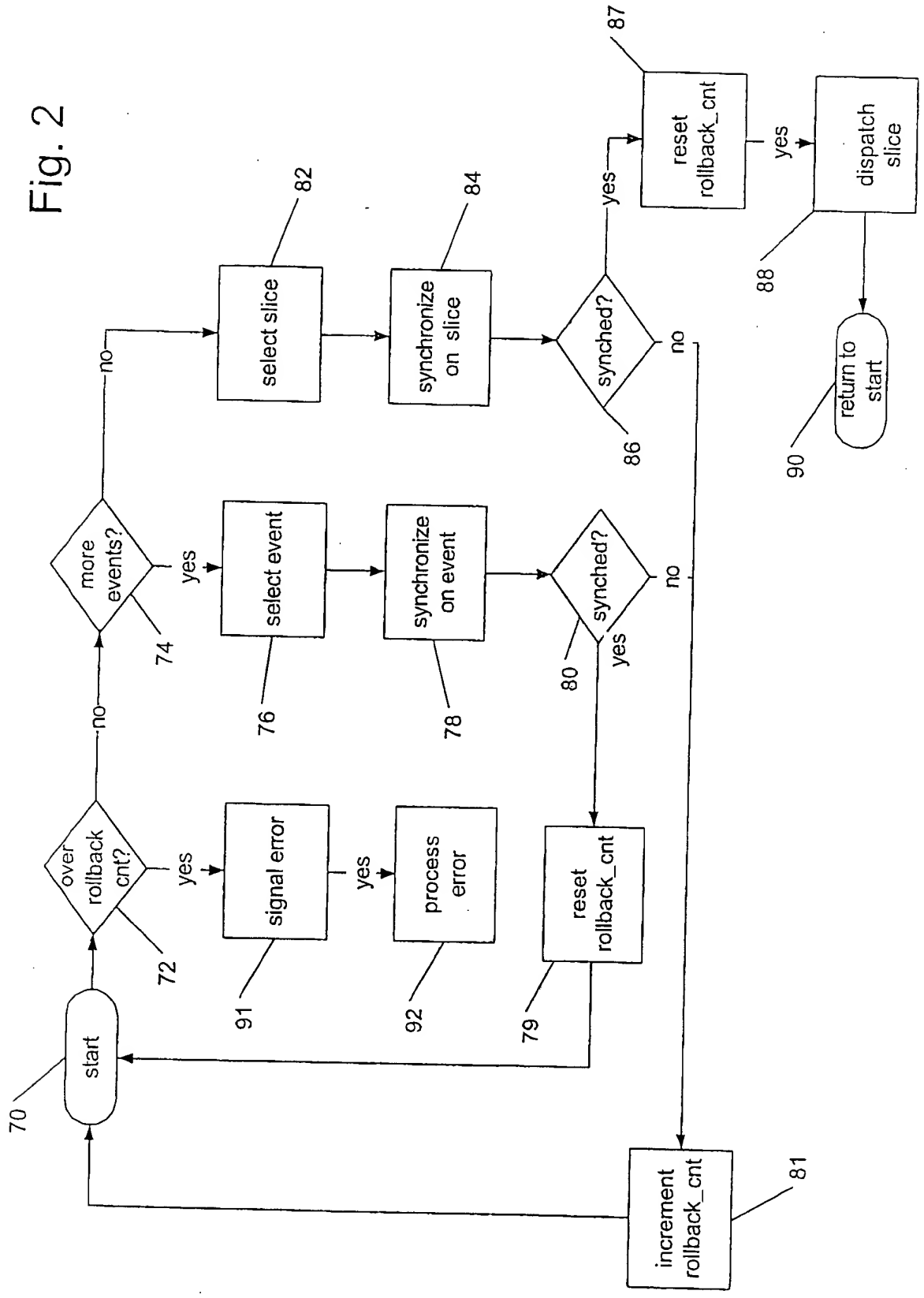

12
32
48
50
52
17
20
Events
Code
16
28
24
Scheduler
Synchronizer
Dispatcher
CPU
40
60
62
64
38
10
54
56
58
18
22
Scheduler
Synchronizer
Dispatcher
CPU
14
26
Events
Code
Network
30
42
44
46

Fig. 1

Fig. 2


start
70
over rollback cnt?
72
yes
signal error
91
yes
process error
92
reset rollback_cnt
79
no
more events?
74
yes
select event
76
synchronize on event
78
synched?
80
yes
reset rollback_cnt
87
yes
dispatch slice
88
return to start
90
no
synched?
86
yes
synchronize on slice
84
select slice
82
no
increment rollback_cnt
81
no

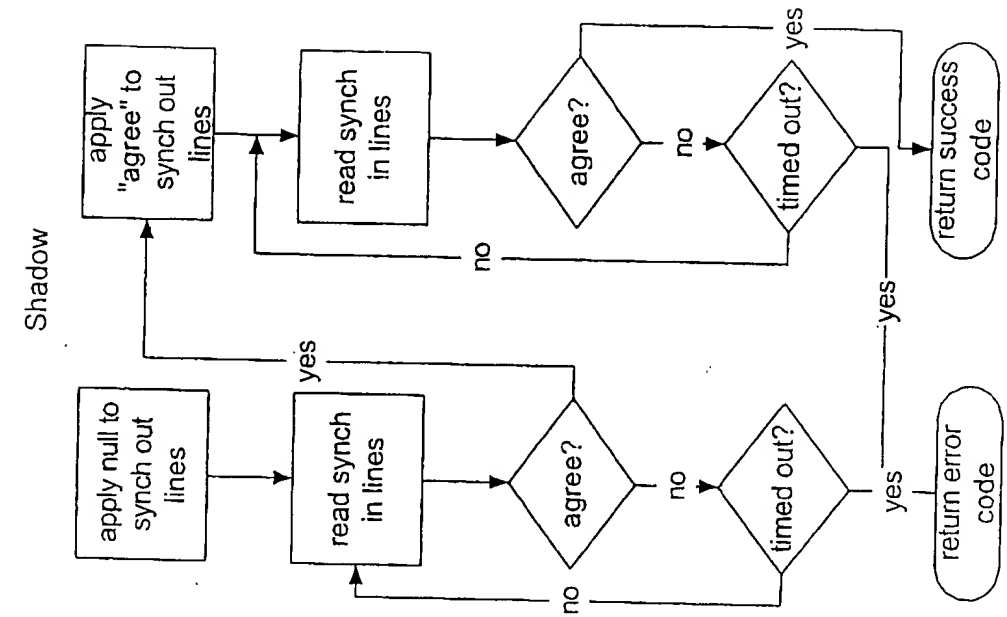

Shadow
apply "agree" to synch out lines
read synch in lines
agree?
no
timed out?
yes
return success code
no
yes
apply null to synch out lines
read synch in lines
agree?
no
timed out?
yes
return error code
no
yes

Fig. 3B

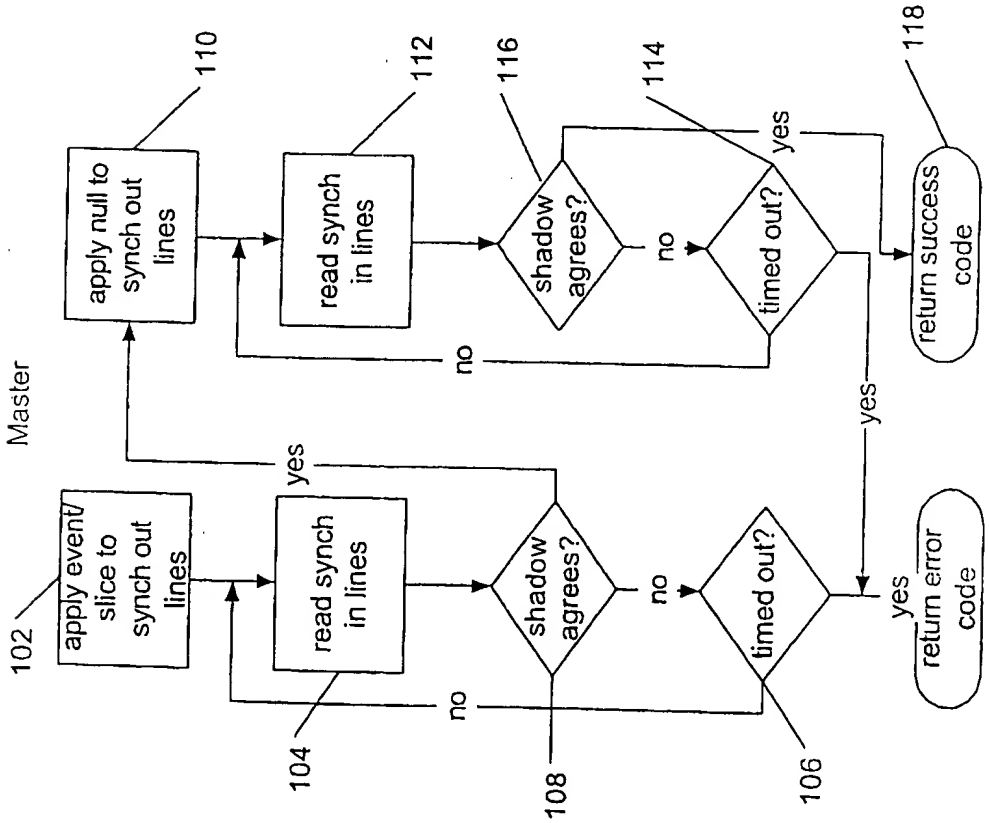

Master
102
apply event/ slice to synch out lines
read synch in lines
104
shadow agrees?
108
no
timed out?
106
yes
return error code
yes
no
yes
apply null to synch out lines
110
read synch in lines
112
shadow agrees?
116
no
timed out?
114
yes
return success code
118
no
yes

Fig. 3A

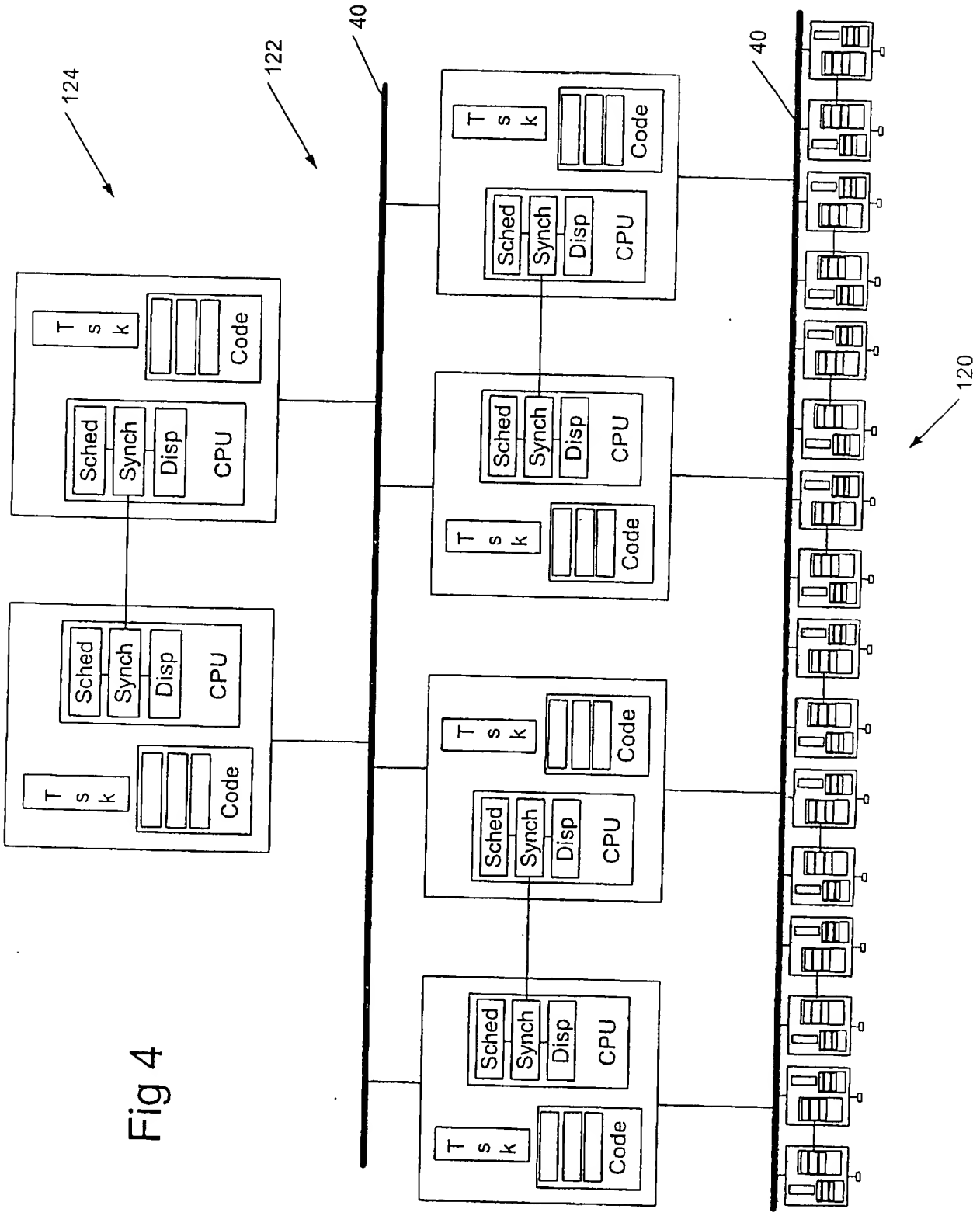


Fig 4